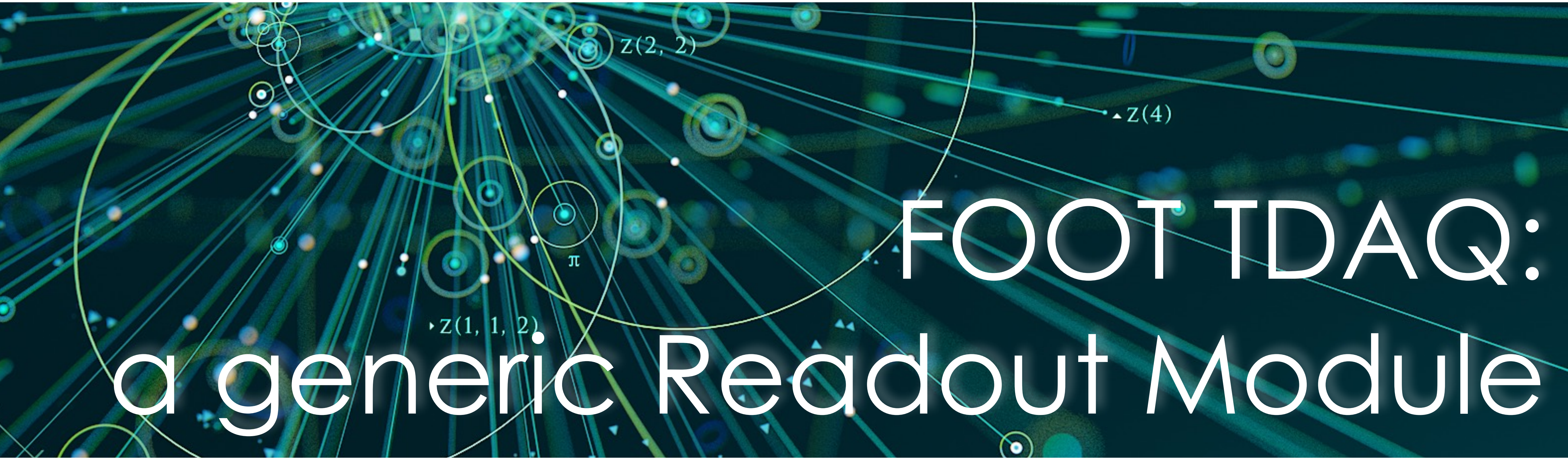


FOOT TDAQ: a generic Readout Module

Silvia Biondi, Mauro Villa

University & INFN of Bologna

FOOT General Meeting - Bologna

04.12.2017

Intro

Goal of the work:

1. interface between the central DAQ and a **remote device**

2. a **flexible software**, able to connect to different devices if needed

Details of the work:

1. **Readout Module** of a generic detector;

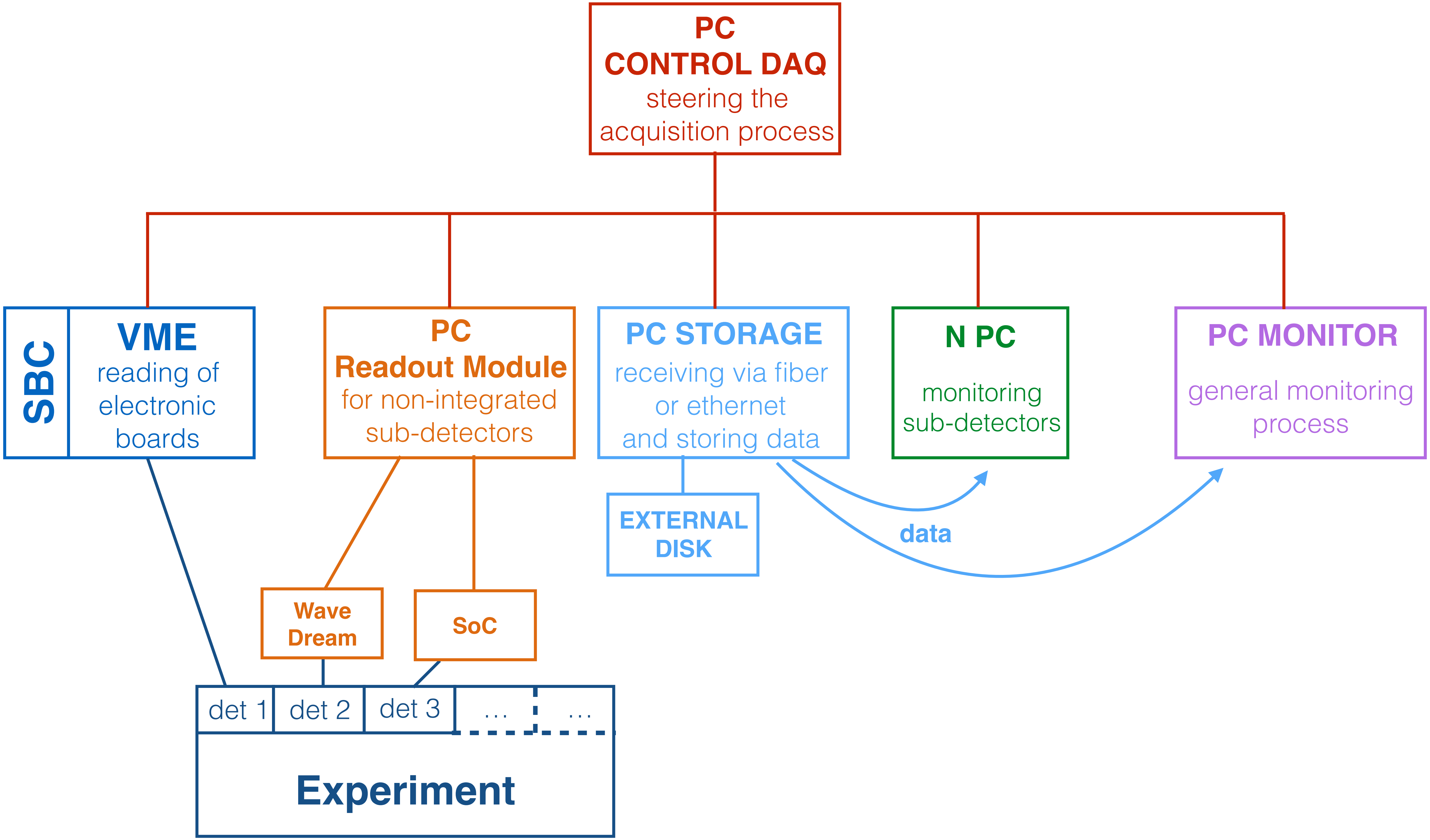
2. connections to a remote device for many purposes:

    1. transmission of **configuration and monitoring parameters**;

    2. flux of **data** from the device;

3. using a **“fake” server** (a simple test macro sending “fake” data to client).

FOOT TDAQ System



PC
CONTROL DAQ
steering the acquisition process

SBC

VME
reading of electronic boards

PC
Readout Module
for non-integrated sub-detectors

PC STORAGE
receiving via fiber or ethernet and storing data

N PC
monitoring sub-detectors

PC MONITOR
general monitoring process

EXTERNAL DISK

data

Wave Dream

SoC

det 1

det 2

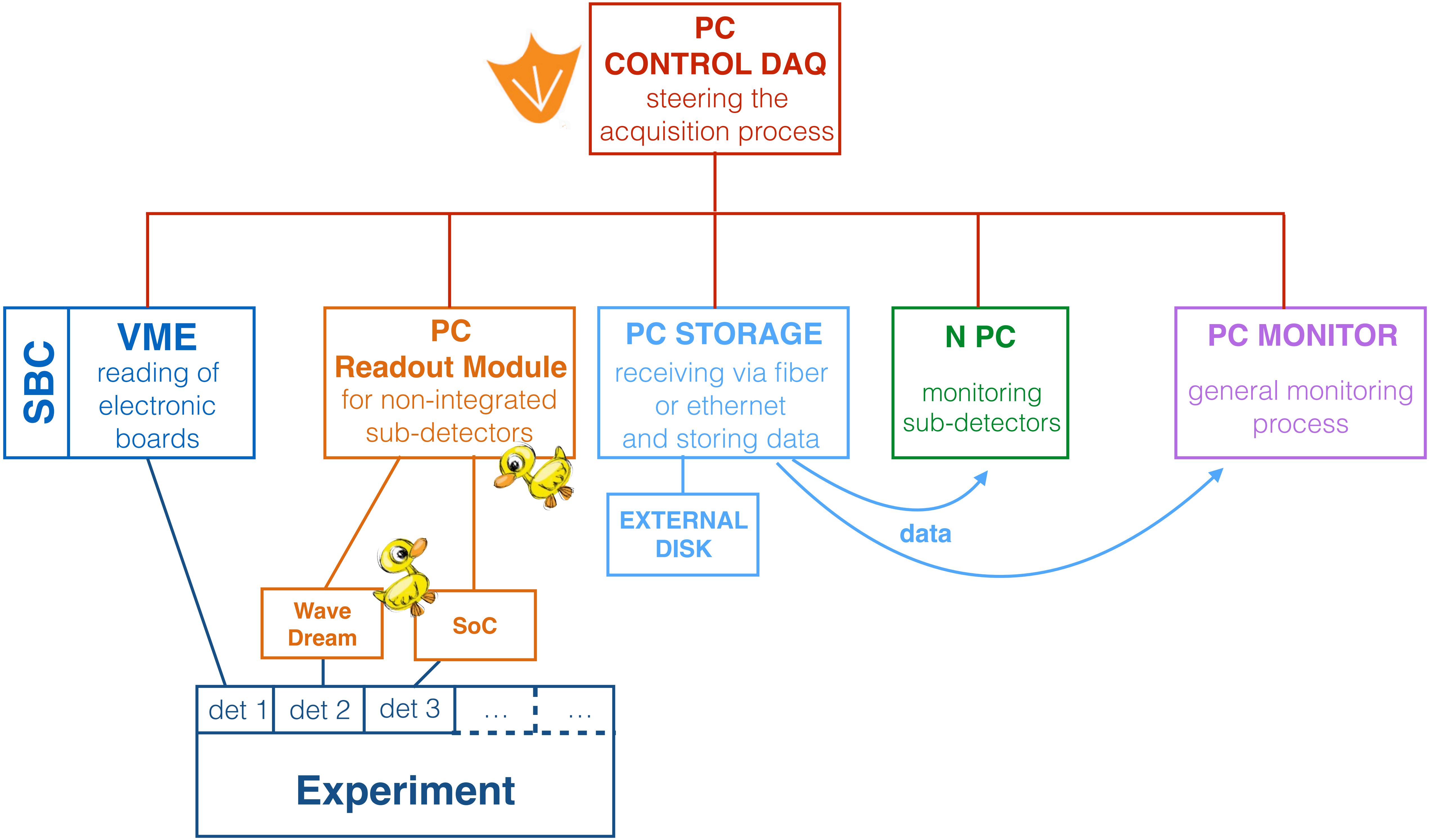
det 3

…

…

Experiment

FOOT TDAQ System



PC CONTROL DAQ
steering the acquisition process

SBC | **VME**
reading of electronic boards

PC Readout Module
for non-integrated sub-detectors

PC STORAGE
receiving via fiber or ethernet and storing data

EXTERNAL DISK

N PC
monitoring sub-detectors

PC MONITOR
general monitoring process

data

Wave Dream

SoC

det 1 | det 2 | det 3 | … | …

Experiment

Code hierarchy

FOOT Partition

- ➡ segment (1 block)
- ➡ segment (1 block, i.e. VME)
    - ➡ RCD/ROS
        - ➡ Readout Module
    - ➡ etc…
- ➡ segment (1 block)
- ➡ etc…

Used tools

- ➡ database OKS
- ➡ C++ classes and objects

Code hierarchy

FOOT Partition

- segment (1 block)
- segment (1 block, i.e. VME)
  - RCD/ROS
    - **Readout Module**
  - etc…
- segment (1 block)
- etc…

the topic of this talk concerns this part of our TDAQ framework.

(that will be seen by each sub-detector)

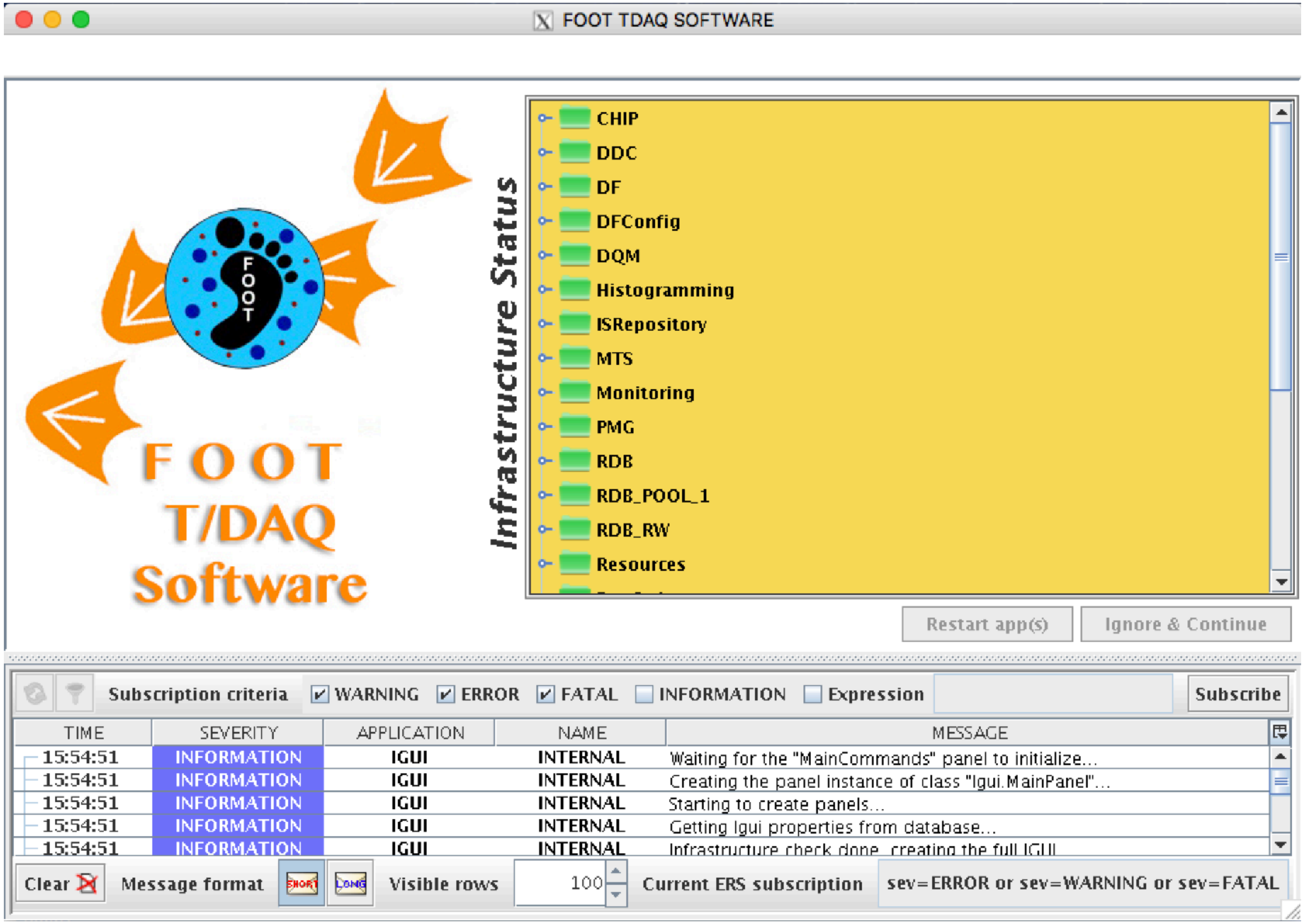
Used tools

- database OKS
- C++ classes and objects

FOOT TDAQ Panel

A FOOT TDAQ Software has been setup

- ➡ able to handle the data stream
- ➡ able to show the significant information of the run



FOOT TDAQ SOFTWARE

- 🟩 CHIP
- 🟩 DDC
- 🟩 DF
- 🟩 DFConfig
- 🟩 DQM
- 🟩 Histogramming
- 🟩 ISRepository
- 🟩 MTS
- 🟩 Monitoring
- 🟩 PMG
- 🟩 RDB
- 🟩 RDB_POOL_1
- 🟩 RDB_RW
- 🟩 Resources

Subscription criteria ☑ WARNING ☑ ERROR ☑ FATAL □ INFORMATION □ Expression

TIME	SEVERITY	APPLICATION	NAME	MESSAGE
15:54:51	INFORMATION	IGUI	INTERNAL	Waiting for the "MainCommands" panel to initialize...
15:54:51	INFORMATION	IGUI	INTERNAL	Creating the panel instance of class "Igui.MainPanel"...
15:54:51	INFORMATION	IGUI	INTERNAL	Starting to create panels...
15:54:51	INFORMATION	IGUI	INTERNAL	Getting Igui properties from database...
15:54:51	INFORMATION	IGUI	INTERNAL	Infrastructure check done, creating the full IGUI

Message format

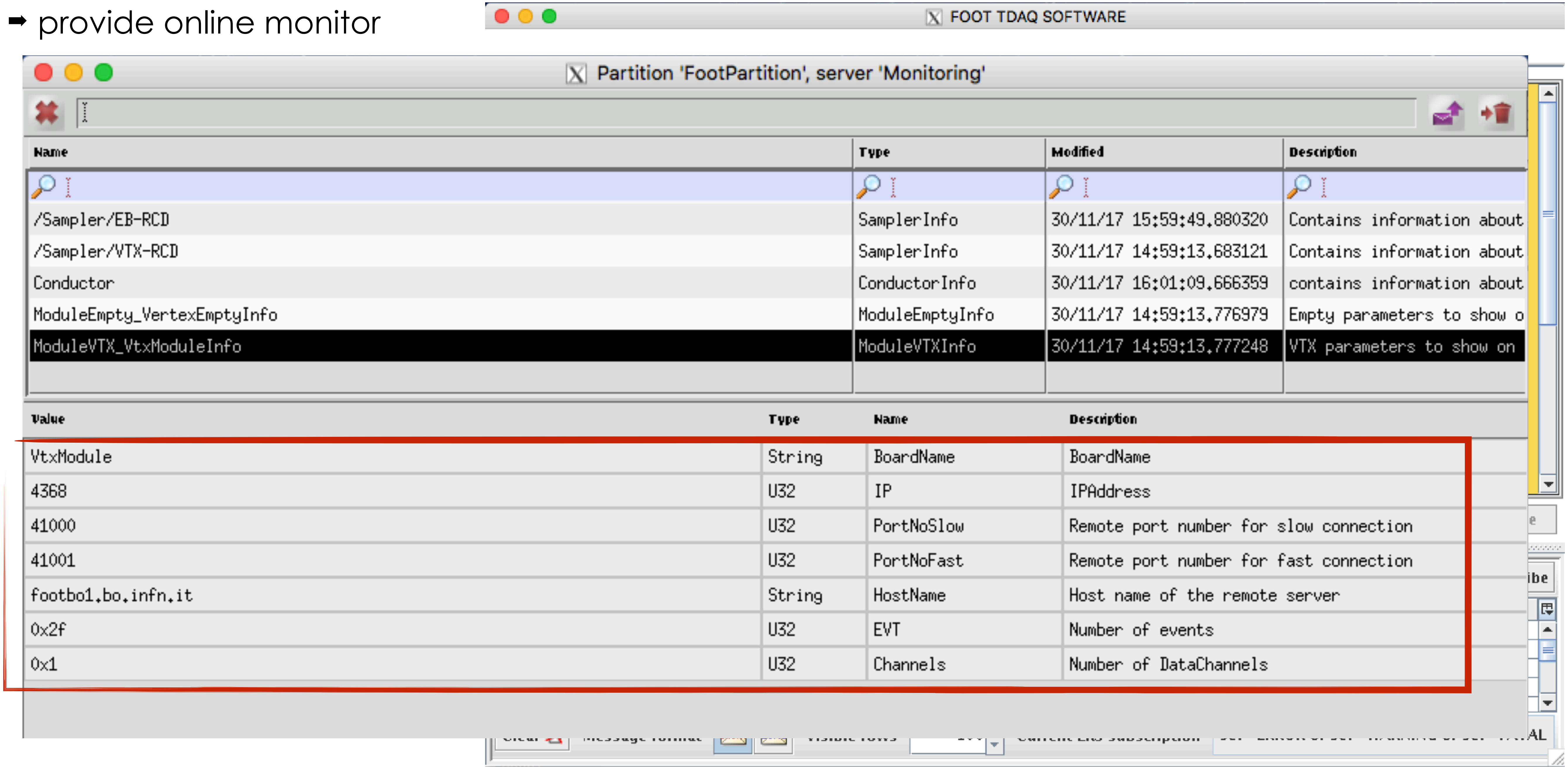
Visible rows 100

Current ERS subscription sev=ERROR or sev=WARNING or sev=FATAL

FOOT TDAQ Panel

A FOOT TDAQ Software has been setup

- ➡ able to handle the data stream
- ➡ able to show the significant information of the run
- ➡ provide online monitor



Name	Type	Modified	Description
/Sampler/EB-RCD	SamplerInfo	30/11/17 15:59:49.880320	Contains information about
/Sampler/VTX-RCD	SamplerInfo	30/11/17 14:59:13.683121	Contains information about
Conductor	ConductorInfo	30/11/17 16:01:09.666359	contains information about
ModuleEmpty_VertexEmptyInfo	ModuleEmptyInfo	30/11/17 14:59:13.776979	Empty parameters to show o
ModuleVTX_VtxModuleInfo	ModuleVTXInfo	30/11/17 14:59:13.777248	VTX parameters to show on

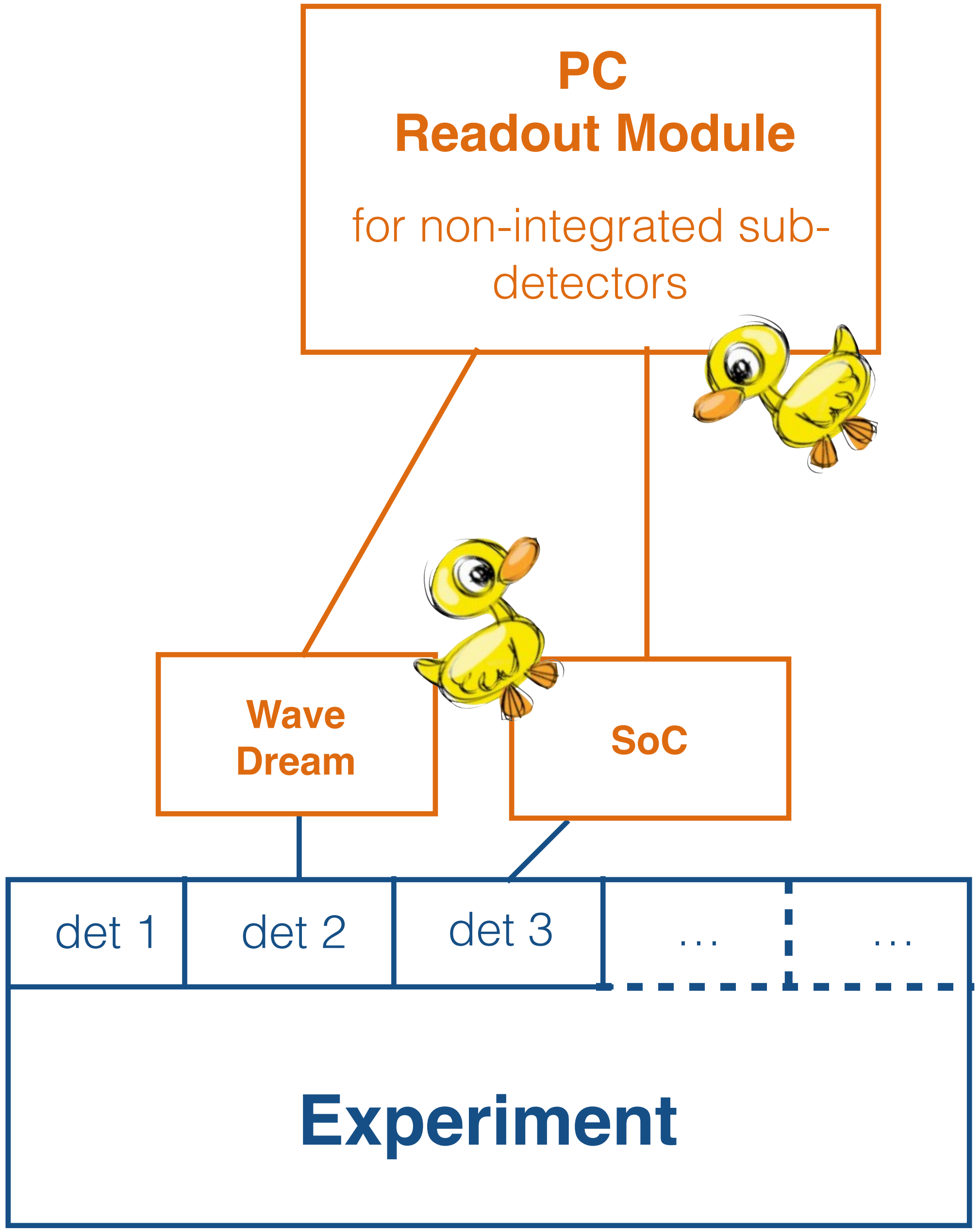
Value	Type	Name	Description
VtxModule	String	BoardName	BoardName
4368	U32	IP	IPAddress
41000	U32	PortNoSlow	Remote port number for slow connection
41001	U32	PortNoFast	Remote port number for fast connection
footbo1.bo.infn.it	String	HostName	Host name of the remote server
0x2f	U32	EVT	Number of events
0x1	U32	Channels	Number of DataChannels

Project idea

SOCKET

↕

SOCKET



PC
Readout Module

for non-integrated sub-detectors

Wave Dream

SoC

det 1

det 2

det 3

…

…

Experiment

CLIENT

↕

SERVER

Tools used in the project

SOCKETS

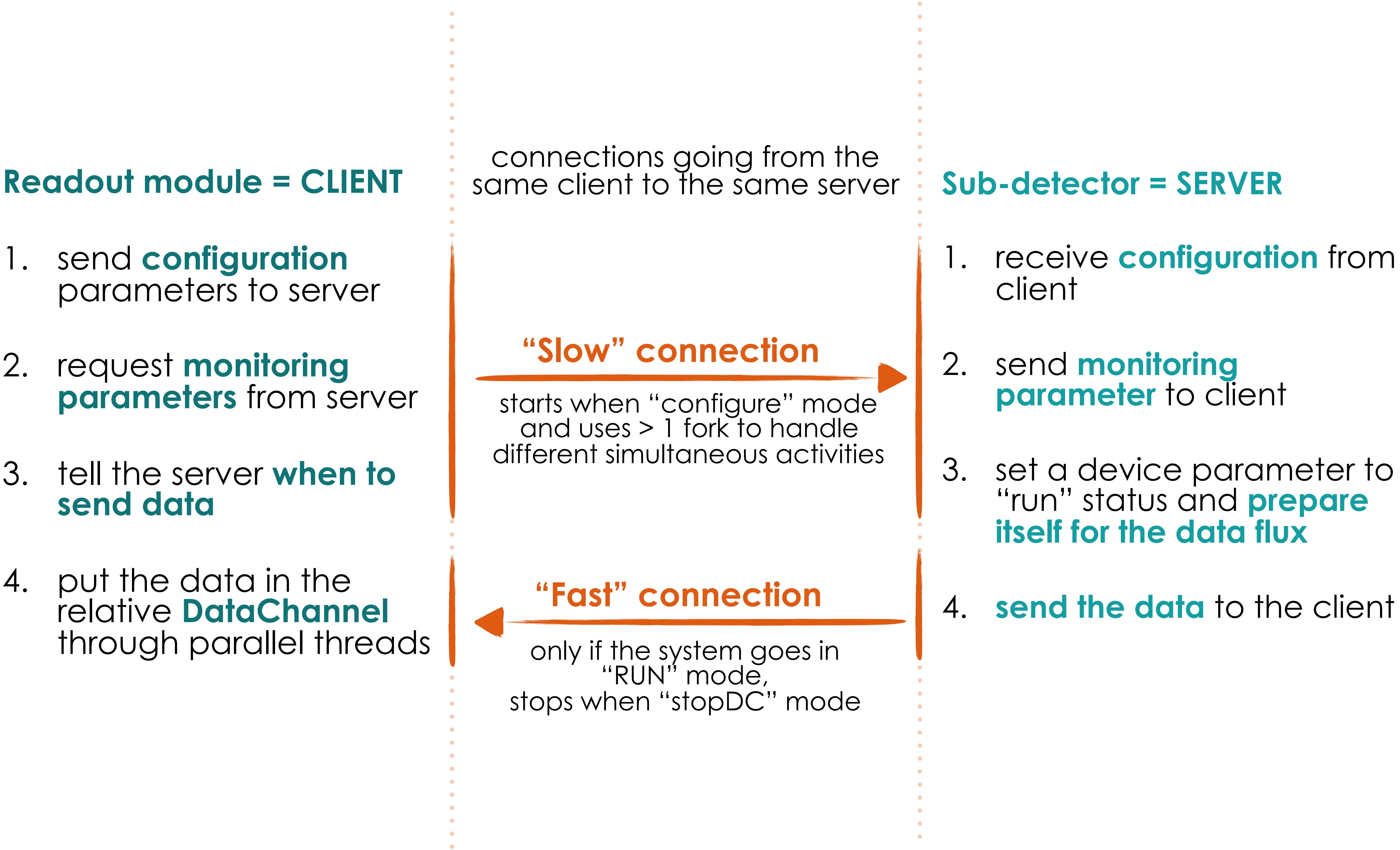
- A way of speaking to other programs using **standard file descriptors**:
  - ➡ make a call to the **socket() system routine**;
  - ➡ after the socket() returns the socket descriptor, start communicate through it using the specialised **send()/recv() socket API calls**.
- Socket uses a **TCP connection**, which is defined by two endpoints (sockets);
- It is the socket pair (the 4-tuple consisting of the **client IP address**, **client port number**, **server IP address** and **server port number**) that specifies the two endpoints that uniquely identifies each TCP connection in an internet;
- The purpose of ports is to **differentiate multiple endpoints** on a given network address.
- **Why using the socket:**
  - ✓ all **C++ Standard Template Library based**;
  - ✓ provide **reliable two-way communication**;
  - ✓ immediate confirmation that **what has been sent actually reached its destination**;
  - ✓ ensure that **data are not lost or duplicated** and that **the order is the same** from the sender to the receiver.

Tools used in the project

CONNECTIONS

- **Transmission Control Protocol/Internet Protocol (TCP/IP)**:
  - ➡ providing **end-to-end communications** that identify how it should be broken into packets, addressed, transmitted, routed and received at the destination.
  - ➡ It uses the **client/server model** of communication in which a user or machine (a client) is provided a service (like sending a message) by another computer (a server) in the network
- **Why using the TCP/IP:**
  - ✓ requires **little central management**;
  - ✓ designed to make **networks reliable**, with the **ability to recover automatically** from the failure of any device on the network;
  - ✓ IP is **compatible with all operating systems and with all types of computer hardware and networks**.

Project scheme



Readout module = CLIENT

1. send **configuration** parameters to server
2. request **monitoring parameters** from server
3. tell the server **when to send data**
4. put the data in the relative **DataChannel** through parallel threads

connections going from the same client to the same server

"Slow" connection

starts when "configure" mode and uses > 1 fork to handle different simultaneous activities

"Fast" connection

only if the system goes in "RUN" mode, stops when "stopDC" mode

Sub-detector = SERVER

1. receive **configuration** from client
2. send **monitoring parameter** to client
3. set a device parameter to "run" status and **prepare itself for the data flux**
4. **send the data** to the client

Future proposal

TOWARDS THE VALIDATION

- Test with the **board** in laboratory:
  - ➡ Linux environment totally outside the TDAQ software.

PROVIDING TO THE COLLABORATION

- a **code skeleton** for both:
  - ✓ client-side: Module in the DAQ software;
  - ✓ server-side: structure to implement in the sub-detector software;
- an **exhaustive documentation** of the code and how to use it (including **assistance**);
- a **git repository** with all the updated material.

Conclusions

SUMMARISING:

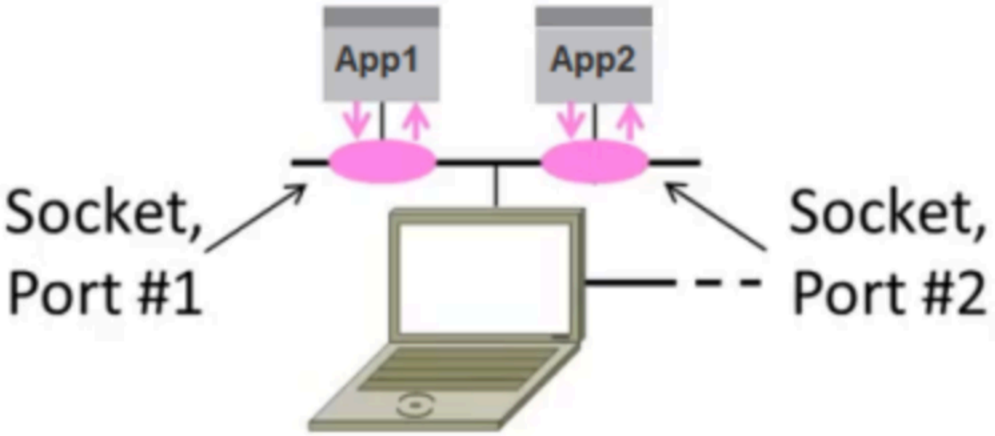
- ✓ **a generic and flexible Readout Module has been set up**;
  - ➡ **reliable two-end-points connection** between TDAQ software and a generic sub-detector;
- ✓ need a **final test with the board** to validate the project;
- ✓ will provide the **skeleton code to implement for each sub-detector**;
- ✓ **full documentation and assistance guaranteed**.

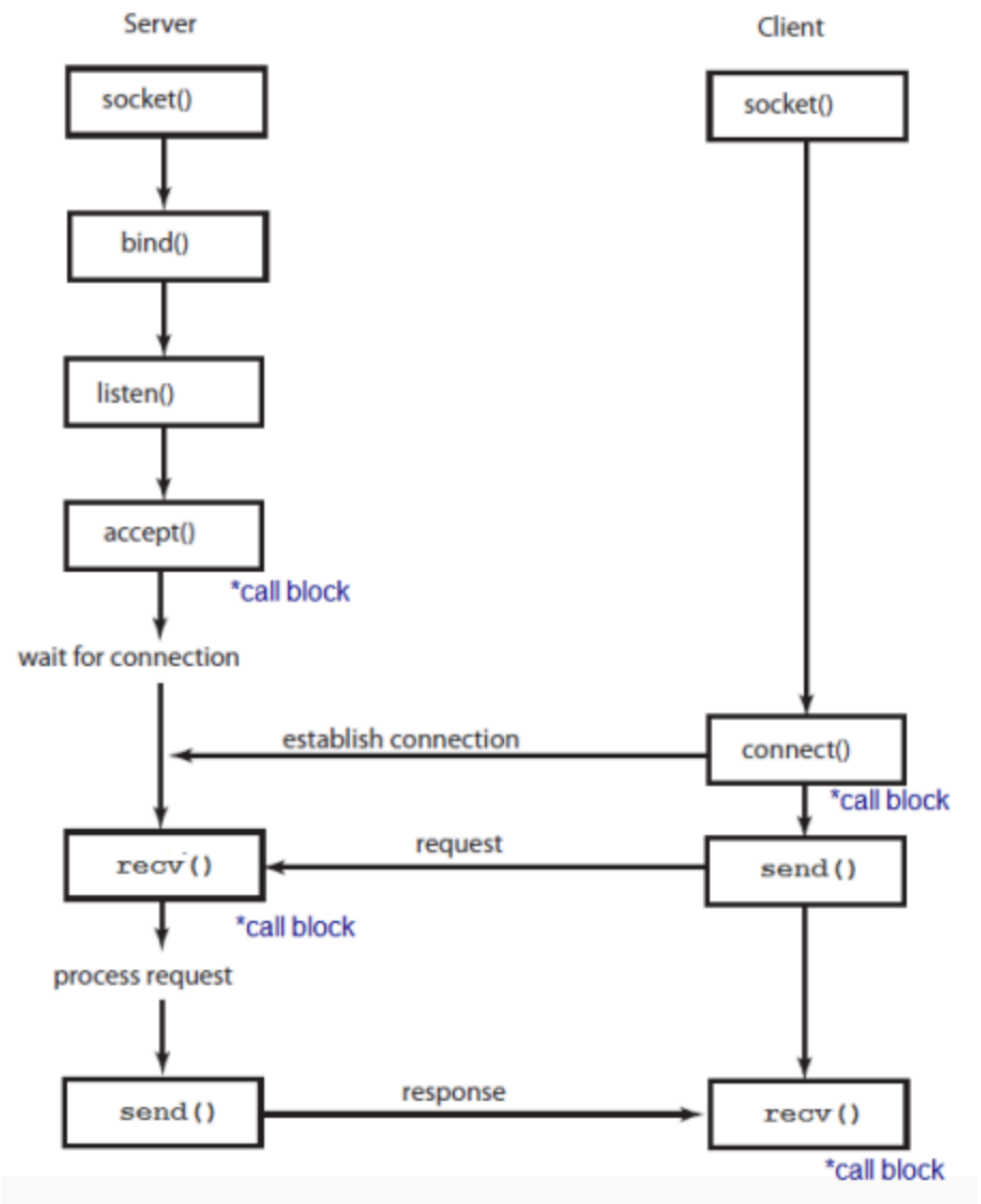
100% RISK FREE ★ GUARANTEE ★ 100% RISK FREE

Supporting material

Sockets and client/server

Sockets let apps attach to the local network at different ports





Server

socket()

bind()

listen()

accept()

*call block

wait for connection

process request

send()

Client

socket()

connect()

*call block

send()

recv()

*call block

establish connection

request

recv()

*call block

response

TCP/IP connection

TCP/IP model layers

- TCP/IP functionality is divided into four layers, each of which include specific protocols.

1. The **application layer** provides applications with standardised data exchange.
2. The **transport layer** is responsible for maintaining end-to-end communications across the network. TCP handles communications between hosts and provides flow control, multiplexing and reliability.
3. The **network layer**, also called the internet layer, deals with packets and connects independent networks to transport the packets across network boundaries.
4. The **physical layer** consists of protocols that operate only on a link - the network component that interconnects nodes or hosts in the network.